(12) UK Patent Application (19) GB (11) 2 344 262 (13) A

(43) Date of A Publication 31.05.2000

(21) Application No 9821524.7

(22) Date of Filing 02.10.1998

(51) INT CL[7]
H04L 12/24 , G06F 17/30

(52) UK CL (Edition R)
H4P PEUX PPG

(56) Documents Cited
EP 0621705 A2

(58) Field of Search
UK CL (Edition R) G4A AMA AMG1 , H4P PEUX PPG
INT CL[7] G06F 17/30 , H04L 12/24 12/26
Online: WPI, EPODOC, JAPIO

(71) Applicant(s)
General Datacomm Inc
(Incorporated in USA - Delaware)
Straits Turnpike, Route 63, Middlebury,
Connecticut 06762-1299, United States of America

(72) Inventor(s)
David Gymer
Paul Burden

(74) Agent and/or Address for Service
Mathys & Squire
100 Grays Inn Road, LONDON, WC1X 8AL,
United Kingdom

(54) Abstract Title
Retrieval of network management information

(57) The invention is concerned with accessing and retrieving information from tables such as a management information base (MIB) using network management protocols, particularly the Simple Network Management (SNMP) protocol. To overcome problems associated with using object identifiers (OIDs) in conventional network management protocols, a modified protocol is disclosed in which some object identifiers are abbreviated or replaced by indices. The protocol data unit (PDU) used to retrieve information comprises an identifier signifying that it is a table block access request, an object identifier of a table to be accessed, an index to a row within the table and information identifying the number of rows to be accessed. This can lead to significant improvements in efficiency compared to multiple "Get Next" operations or "Get Bulk" operations (see figures 1-3), with only a small modification to the protocol.

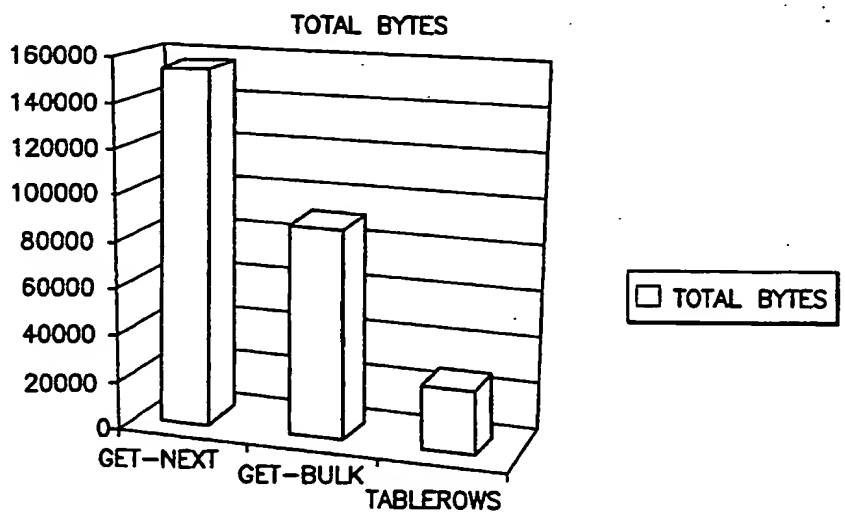

FIG.1

GB 2 344 262 A

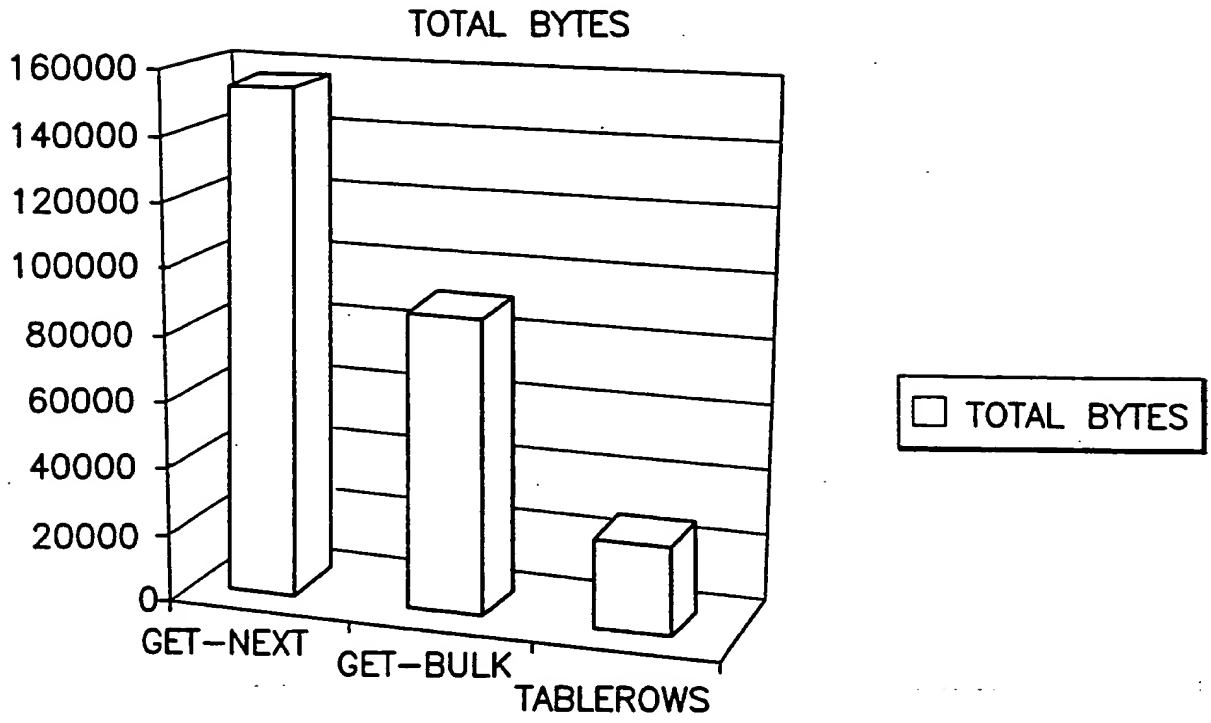
TOTAL BYTES
160000
140000
120000
100000
80000
60000
40000
20000
0
GET-NEXT
GET-BULK
TABLEROWS
TOTAL BYTES

FIG.1

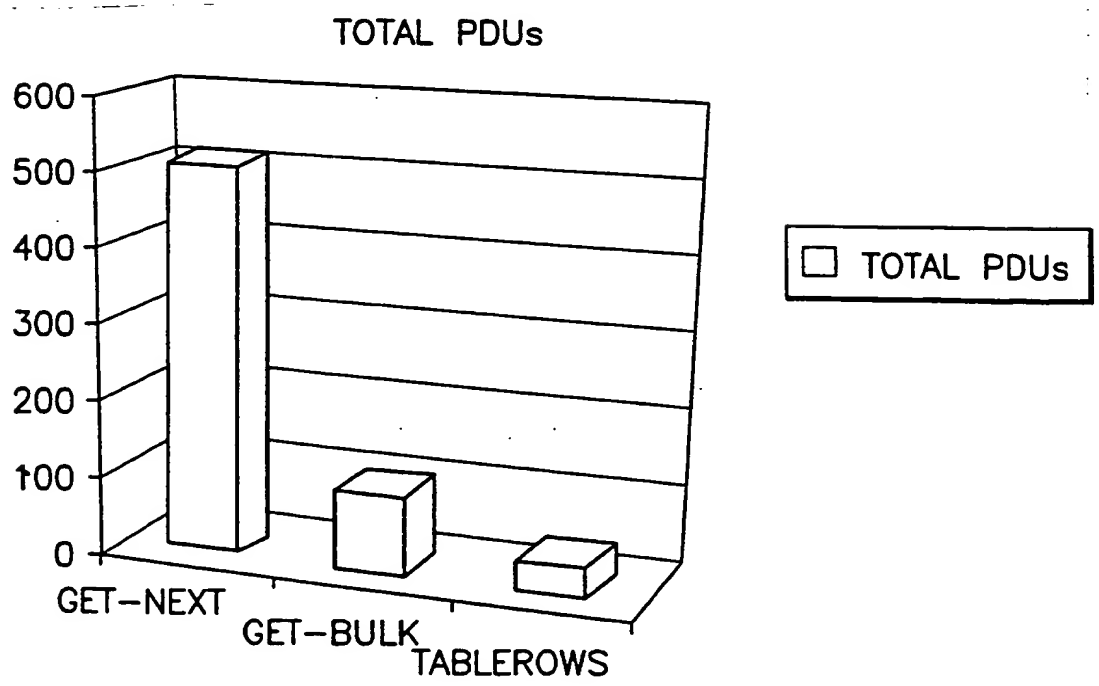
TOTAL PDUs
600
500
400
300
200
100
0
GET-NEXT
GET-BULK
TABLEROWS
TOTAL PDUs

FIG.2

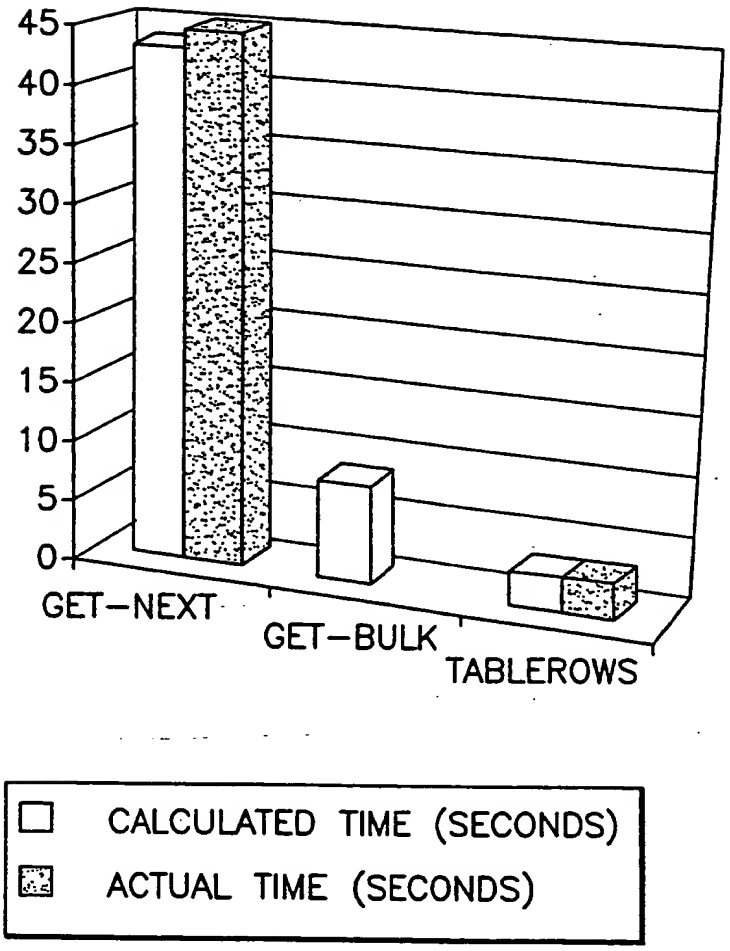
45
40
35
30
25
20
15
10
5
0
GET–NEXT
GET–BULK
TABLEROWS
CALCULATED TIME (SECONDS)
ACTUAL TIME (SECONDS)

FIG.3

by the term. Devices which are responsive to a subset or derivative of SNMP commands are intended to be encompassed by the invention.

Another advantage is that the Object Identifiers of the rows and objects within the table need not be communicated in the response packet; preferably Object Identifiers are only communicated in the response packet if specifically requested. Preferably, if Object Identifiers for the rows are requested, a single Object Identifier, preferably abbreviated, is communicated for each row. It is well- known that Object Identifiers are hierarchical, the Object Identifier of an item within a table comprising the Object Identifier of the table with suffixes dependent on the row and column within the table. By "abbreviated" is meant sufficient identification information from the suffixes, optionally pre-pended with a further portion of the complete Object Identifier or a dummy prefix, but not including the entire Object Identifier.

Preferably, information representative of the number of rows actually included in the response packet is included in the response packet, at least when the number of rows supplied differs from the number of rows requested. This may facilitate determination by the requestor of the amount of information supplied and composition of a subsequent request for remaining information.

Preferably, the method includes selecting one or more columns from which data is to be included based on column identifier information within the received Protocol Data Unit. This may allow data to be selectively extracted from multiple columns and multiple rows within a single operation. Most preferably, the column identifier information is in the form of index information. This avoids the need to communicate the Object Identifier to each column, and allows specified columns to be accessed even when the Object Identifiers are not known.

In a second aspect, the invention provides a method, in a network

management device which issues and accepts network management protocol, preferably Simple Network Management Protocol, Protocol Data Units, of obtaining data from a table in a remote device, preferably arranged to perform a method as defined above, the method comprising:

determining:- (a) an Object Identifier of a table in the remote device to be accessed;

(b) an index to the start of a block of rows from which data within the table is required;

(c) the number of rows to be accessed;

composing a Protocol Data Unit designated as a table block access request and including information representative of said determining;

outputting the Protocol Data Unit to the remote device; and

obtaining said data from a response Protocol Data Unit received from the remote device.

Preferably, the method further comprises determining whether the received Protocol Data Unit contains all the information requested and, if not, composing a further request for information.

The method may further comprise supplying the information to a management application.

In a third aspect, the invention provides a network device comprising:

means for responding to Protocol Data Units received containing network management protocol, preferably Simple Network Management Protocol, commands;

means for identifying a received Protocol Data Unit designated as a table block access request;

means for indexing a portion of a stored table based on an Object Identifier and an index to a row to be read from the table from the Protocol Data Unit;

means for determining the number of rows to be read based on

information obtained from the Protocol Data Unit;

means for looking up information in the table based on the Object Identifier and the index to the row to be read;

means for composing a response Protocol Data Unit containing information read from the table for a plurality of rows based on the number of rows to be read.

According to a fourth aspect, the invention provides a Protocol Data Unit comprising:

an identifier signifying that the Protocol Data Unit is a table block access request;

an Object Identifier of a table to be accessed;

an index to a row within the table to be accessed;

information identifying the number of rows to be accessed.

The Protocol Data Unit preferably further comprises information identifying the number of columns in the table to be accessed and an identifier for each column.

It will be appreciated that the invention can be applied regardless of the information contained within the table to the access and provide a technical improvement in terms of more efficient data transfer and simplified access to large tables.

An embodiment of the invention will now be described, by way of example, with reference to the accompanying drawings in which:

Fig. 1 is a graph illustrating a comparison between the amount of data to be transferred when access a large table according to conventional methods and according to an embodiment of the invention;

Fig. 2 is a graph illustrating a comparison between the amount of Protocol Data Units to be transferred when access a large table according to conventional methods and according to an embodiment of the invention;

Fig. 3 is a graph illustrating a comparison between the amount of time taken for table retrieval when access a large table according to conventional methods and according to an embodiment of the invention.

An embodiment for use in an SNMP-compatible network device having a plurality of MIB tables stored therein will now be described. Details of conventional MIB tables and SNMP commands, together with details of Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER) encoding are assumed to be well-known and will not be described in detail; reference should be made to The Simple Book, together with references 40-53 in the bibliography thereon, or to any of the relevant standards, all of which are incorporated herein by reference.

By way of background summary information, basic formats of an SNMP message, a generic PDU, a request PDU, a Get PDU and a Get Next PDU will be set out, in ASN.1 syntax.

Firstly, a basic message format:-

```
-- top-level message

   Message ::=
          SEQUENCE {
              version        -- version-1 for this RFC
                 INTEGER {
                     version-1(0)
                 },

              community      -- community name
                 OCTET STRING,

              data           -- e.g., PDUs if trivial
                 ANY         -- authentication is being used
          }
```

Next, the format of a Protocol Data Unit:-

```
-- protocol data units

      PDUs ::=
             CHOICE {
                 get-request
                     GetRequest-PDU,

                 get-next-request
                     GetNextRequest-PDU,

                 get-response
                     GetResponse-PDU,
```

```
            set-request
                SetRequest-PDU,

            trap
                Trap-PDU
            }

    -- the individual PDUs and commonly used
    -- data types will be defined later

    END
```

The basic format of a request PDU will now be set out:-

```
    -- request/response information

    RequestID ::=
            INTEGER

    ErrorStatus ::=
            INTEGER {
                noError(0),
                tooBig(1),
                noSuchName(2),
                badValue(3),
                readOnly(4)
                genErr(5)
            }

    ErrorIndex ::=
            INTEGER

    -- variable bindings

    VarBind ::=
            SEQUENCE {
                name
                    ObjectName,

                value
                    ObjectSyntax
            }

    VarBindList ::=
            SEQUENCE OF
                VarBind
```

The format of a standard "Get" PDU is:-

```
    GetRequest-PDU ::=
        [0]
            IMPLICIT SEQUENCE {
                request-id
                    RequestID,

                error-status        -- always 0
                    ErrorStatus,

                error-index         -- always 0
                    ErrorIndex,

                variable-bindings
                    VarBindList
            }
```

The format of a "Get Next" PDU is:-

```
GetNextRequest-PDU ::=
    [1]
        IMPLICIT SEQUENCE {
            request-id
                RequestID.

            error-status        -- always 0
                ErrorStatus.

            error-index         -- always 0
                ErrorIndex.

            variable-bindings
                VarBindList
        }
```

Further details of the components of the entities defined above and other background information may be found by reference to RFC 1157 or other standard texts.

According to this embodiment, we propose a modified PDU which we designate a Get Table Row message. This is defined below using the ASN.1 syntax:-

```
GetTableRow-PDU ::=
        SEQUENCE {
            request-id
                INTEGER.

            error-status
                INTEGER {
                    noError(0).
                    tooBig(1).
                    noSuchName(2).
                    badValue(3).
                    readOnly(4).
                    genErr(5)
                }.

            error-index
                INTEGER.
            snmpp-version
                INTEGER {
                    version1 (1)                    -- First implementation
                }.

            table-name
                OBJECT-INDENTIFIER.    -- OID of the table being retrieved
            start-index
                INTEGER.                            -- starting row index for retrieval
            max-rows
                INTEGER.                            -- maximum no. of rows to be retrieved
                                                    -- -1 indicates "get all rows"
            table-size
                INTEGER.                            -- No. of rows in table

            instances-included
                INTEGER {
                    no(0),                          -- Row instances not encoded
                    yes(1)                          -- Row instances are encoded
                }.

            column-total
                INTEGER.                            -- No.of columns to be retrieved

            column1
                INTEGER.                            -- column id for first column
            column2
                INTEGER.                            -- column id for second column
            ...
            columnN
                INTEGER.                            -- column id for Nth column.

            variable-bindings
                varBindList
        }
```

```
VarBind ::=
        SEQUENCE {
                row-instance'
                value           OBJECT-IDENTIFIER,      -- optional instance OID for row
                                objectSyntax                    -- value for this row/column entry
        }
```

This command is intended to allow a management application to retrieve arbitrary rows from a table without having to issue repeated GetNext commands to get to the correct rows. For optimum efficiency and flexibility, it is found to be highly desirable that the command can access arbitrary columns, and not just complete rows.

An explanation of the fields in a **GetTableRows** request PDU as would be sent from a management application follows:-

- **request-id**
  The unique request id for this PDU
- **snmpp-version**
  Indicates the revision level of the SNMPP PDU (should always be set to 1).
- **table-name**
  The OBJECT IDENTIFIER representing the table to be retrieved. For example, the interfaces table in rfc1213 would have a table name of **1.3.6.1.2.1.2.2**
- **start-index**
  Identifies the first row index to be retrieved from the table. This represents essentially the row number in that table (starting 0). So, to start retrieving from the first row, start-index would be set to 0. To retrieve from the 25th row, start-index would be set to 24, etc.
- **max-rows**
  Represents the maximum number of rows to be retrieved (if possible). If all rows from the **start-index** to end of table are required, this should be set to -1.
- **column-total**
  Represents the total number of columns to be retrieved from the table (the column ids are encoded immediately after this object in the PDU).

column-id

A column id is encoded for each of the columns requested. So, for example, if five columns had been requested, then five consecutive INTEGERS would be encoded representing the respective column ids. The id represents the conceptual column number for that table (starting 1). So, for example, consider the **ifTable** of rfc1213, the column-id for **ifOperStatus** would be 8, since this is the eighth conceptual column in the table.

The request PDU will contain an empty varbind list (since all the information above is sufficient to identify what we are requesting).

Note: All the other objects exist in the request PDU, but will have their default values set.

To implement this embodiment, the (modified) SNMP agent of the network device must process an incoming **GetTableRows** request and package the response message to send back to the requestor. The agent should attempt to include all the requested rows into the response PDU, but due to the restrictions of message size, this may not be possible. In these cases, it should send back as many rows as it can, updating the associated fields to identify precisely the rows it has returned (this is so that the requestor can send another **GetTableRows** request message amended to retrieve the remaining rows).

A **GetTableRows** response PDU should be sent to the management application with the following fields set:-

request-id

The unique request id for this PDU.

snmpp-version

Indicates the revision level of the SNMPP PDU (should always be set to 1)

- **table-name**
  The OBJECT IDENTIFIER representing the table to be retrieved. For example, the interfaces table in rfc1213 would have a table name of **1.3.6.1.2.1.2.2**. This must match the request PDU.
- **start-index**
  Identifies the first row index to be retrieved from the table. This represents essentially the row number in that table (starting 0). So, to start retrieving from the first row, start-index would be set to 0. To retrieve from the 25th row, start-index would be set to 24, etc. This must match the request PDU.
- **max-rows**
  This will be set to the actual number of rows included in this response PDU.
- **table-size**
  Stores the actual size of the table requested (i.e. how many rows exist in the table at that point in time).
- **instances-included**
  set to **no(0)** if the row instances have not been encoded in the varbinds representing the first column requested, otherwise set to **yes(1)** if they have.
- **column-total**
  Represents the total number of columns retrieved from the table (the column ids are encoded immediately after this object in the PDU). This must match the request PDU.
- **column-id**
  A column id is encoded for each of the columns requested. So, for example, then five consecutive INTEGERS would be encoded representing the respective column ids. The id represents the conceptual column number for that table (starting 1). So, for example, consider the **ifTable** of rfc1213, the column-id for **ifOperStatus** would be 8, since this is the eighth conceptual column in the table. Each of these column-ids must match the request PDU.

- **varbind list**

  A list of varbinds must be encoded which represent the data contained in the rows returned. The order of the varbind list is on a per-row basis. So, for example, if five columns had been requested, the first five varbinds would constitute the values for the first row returned, where varbind1 represents the data for column1, varbind2 contains the data for column2 and so on. In most cases, the nam of the varbind is **not** encoded (see the later section on varbind encoding).

The **SNMPP GetTableRows** message is encoded with a message type of 0xAF, which corresponds to:-

ASN_CONTEXT | ASN_CONSTRUCTOR | 0Xf

A variable binding list returned in a **GetTableRows** response message will contain each of the values within the table encoded as usual varbind objects. The varbind list must always contain enough variables encoded in the varbind list will be multiples of **column-total**.

The variable binding for each element in a row will be encoded in order of column-ids requested. The object-name of a varbind will only be encoded if the following two criteria are met:-

1. The **instances-included** variable is set to **yes(1)**
2. The varbind being encoded represents the first column-id of a row.

If the object name is encoded, it will represent the instance oid identifying that row (starting with 0.0, because the first two subids must each be encoded in a single octet according to SNMP).

This is best explained by example, so consider the **ifTable** and the **TableRows** request message has requested two columns, namely

ifAdminStatus(1.3.6.1.2.1.2.2.1.7) and **ifOperStatus**(1.3.6.1.2.1.2.2.1.8).

The column-ids will be encoded as two INTEGERs, namely **7** and **8**.

Supposing the response message was returning 3 rows (for ifIndex 1,2 and 3). The varbind list will be encoded as follows:-

Varbind	Object Name (row-Instance)	Value
1	0.0.1	up(1)
2	Not Encoded	up(1)
3	0.0.2	up(1)
4	Not Encoded	down(2)
5	0.0.3	testing(3)
6	Not Encoded	Down(2)

The above varbinds would represent the following three rows in the **ifTable**:-

ifIndex	ifAdminStatus	ifOperstatus
1	up(1)	up(1)
2	up(2)	down(2)
3	testing(3)	down(2)

The following pseudo-code outlines the basic steps to be performed to implement the embodiment (some of which will co-exist with other steps which are part of a conventional SNMP agent) :-

```
- Receive PDU
[Other SNMP processing]
- Check whether PDU designated "GetTableRows"
```

```
- If not so designated, skip to Continued Processing
- If so designated:-
          - Obtain OID of table to be read from table-name
          - Obtain index to first row to read from start-index
          - Obtain number of rows to read from max-rows
          - Obtain indices to columns to be read column1..N
          - Check whether encoded row ids requested in instances-included
          - Look up information in specified table using indices
          - Compose response packet including:-
                    * Information read from table in varbinds
                    * Number of rows actually read in max-rows
                    * Row ids if specified in varbinds for first column
          - Output response packet
[Continued Processing]
```

It will be appreciated that the ordering of information is not critical and can be changed, as can all labels used both for entities with the PDU and the PDU designation (the label GetTableRows being used here as a suitable label to designate a table block access request). The information contained in the PDU may be replaced by other combinations of information which achieve the same function (for example, the last row may be supplied in place of the first row, and the indexing performed in reverse). Not all functions need be included.

Each feature described above may be provided independently, unless otherwise stated.

Claims:

1. A method of supplying data from a table in a device which is responsive to network management protocol commands, the method comprising receiving a Protocol Data Unit designated as a table block access request;

identifying the Protocol Data Unit as a table block access request;

obtaining an Object Identifier of a table to be read from the Protocol Data Unit;

obtaining an index to a row to be read from the table from the Protocol Data Unit;

determining the number of rows to be read based on information obtained from the Protocol Data Unit;

looking up information in the table based on the Object Identifier and the index to the row to be read;

composing a response Protocol Data Unit containing information read from the table for a plurality of rows based on the number of rows to be read;

outputting the response packet.

2. A method according to Claim 1, wherein Object Identifiers are only included in the response packet if requested.

3. A method according to Claim 1 or Claim 2, wherein if Object Identifiers for the rows are to be included in the response packet, a single Object Identifier is included for each row.

4. A method according to Claim 2 or Claim 3 wherein abbreviated Object Identifiers are included in the response packet.

5. A method according to any preceding claim wherein information representative of the number of rows actually included in the response

packet is included in the response packet, at least when the number of rows supplied differs from the number of rows requested.

6. A method according to any preceding claim including selecting one or more columns from which data is to be included based on column identifier information within the received Protocol Data Unit.

7. A method according to Claim 6, wherein the column identifier information is in the form of index information.

8. A method, in a network management device which issues and accepts network management protocol Protocol Data Units, of obtaining data from a table in a remote device, preferably arranged to perform a method according to any preceding claim, the method comprising:

determining:- (a) an Object Identifier of a table in the remote device to be accessed;

(b) an index to the start of a block of rows from which data within the table is required;

(c) the number of rows to be accessed;

composing a Protocol Data Unit designated as a table block access request and including information representative of said determining;

outputting the Protocol Data Unit to the remote device; and

obtaining said data from a response Protocol Data Unit received from the remote device.

9. A method according to Claim 8 further comprising determining whether the received Protocol Data Unit contains all the data requested and, if not, composing a further request for data.

10. A method according to Claim 8 or Claim 9 further comprising supplying the data to a management application.

11. A method according to any preceding claim, wherein the network management protocol is Simple Network Management Protocol, or a derivative or modification thereof.

12. A network device comprising:

means for responding to Protocol Data Units received containing network management protocol commands;

means for identifying a received Protocol Data Unit designated as a table block access request;

means for indexing a portion of a stored table based on (a) an Object Identifier and (b) an index to a row to be read from the table, obtained from the Protocol Data Unit;

means for determining the number of rows to be read based on information obtained from the Protocol Data Unit;

means for looking up information in the table based on the Object Identifier and the index to the row to be read;

means for composing a response Protocol Data Unit containing information read from the table for a plurality of rows based on the number of rows to be read.

13. A device according to Claim 12, wherein the network management protocol is Simple Network Management Protocol, or a derivative or modification thereof.

14. A Protocol Data Unit comprising:

an identifier signifying that the Protocol Data Unit is a table block access request;

an Object Identifier of a table to be accessed;

an index to a row within the table to be accessed;

information identifying the number of rows to be accessed.

15. A Protocol Data Unit according to Claim 12 further comprising

information identifying the number of columns in the table to be accessed and an identifier for each column.

16. A method of accessing data substantially as herein described.

17. A Protocol Data Unit substantially as herein described.

The Patent Office

INVESTOR IN PEOPLE

20

Application No:	GB 9821524.7	**Examiner:**	Matthew Nelson
Claims searched:	1-17	**Date of search:**	20 March 2000

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.R): G4A (AMA, AMG1); H4P (PEUX, PPG)

Int Cl (Ed.7): G06F 17/30; H04L 12/24, 12/26

Other: Online: WPI, EPODOC, JAPIO

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
A	EP 0621705 A2 (IBM) See col. 2, line 29 - col. 3, line 28 and col. 8, lines 8-29.	

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.